A Fast Fixed-Point Algorithm for Independent Component Analysis

Aapo Hyvärinen and Erkki Oja
Helsinki University of Technology
Laboratory of Computer and Information Science
Rakentajanaukio 2C, 02150 Espoo, Finland
aapo.hyvarinen@hut.fi, erkki.oja@hut.fi

This paper will appear in Neural Computation, 9:1483-1492, 1997.

Abstract

We introduce a novel fast algorithm for Independent Component Analysis, which can be used for blind source separation and feature extraction. It is shown how a neural network learning rule can be transformed into a fixed-point iteration, which provides an algorithm that is very simple, does not depend on any user-defined parameters, and is fast to converge to the most accurate solution allowed by the data. The algorithm finds, one at a time, all non-Gaussian independent components, regardless of their probability distributions. The computations can be performed either in batch mode or in a semi-adaptive manner. The convergence of the algorithm is rigorously proven, and the convergence speed is shown to be cubic. Some comparisons to gradient based algorithms are made, showing that the new algorithm is usually 10 to 100 times faster, sometimes giving the solution in just a few iterations.

1 Introduction

Independent Component Analysis (ICA) (Comon, 1994; Jutten and Herault, 1991) is a signal processing technique whose goal is to express a set of random variables as linear combinations of statistically independent component variables. Two interesting applications of ICA are blind source separation and feature extraction. In the simplest form of ICA (Comon, 1994), we observe m scalar random variables $v_1, v_2, ..., v_m$ which are assumed to be linear combinations of n *unknown independent components* $s_1, s_2, ..., s_n$ that are mutually statistically independent, and zero-mean. In addition, we must assume $n \leq m$. Let us arrange the observed variables v_i into a vector $\mathbf{v} = (v_1, v_2, ..., v_m)^T$ and the component variables s_i into a vector $\mathbf{s}$, respectively; then the linear relationship is given by

$$\mathbf{v} = \mathbf{A}\mathbf{s} \tag{1}$$

Here, $\mathbf{A}$ is an unknown $m \times n$ matrix of full rank, called the mixing matrix. The basic problem of ICA is then to estimate the original components s_i from the mixtures v_j or, equivalently, to estimate the mixing matrix $\mathbf{A}$. The fundamental restriction of the model is that we can only estimate non-Gaussian independent components (except if just one of the independent components is Gaussian). Moreover, neither the energies nor the signs of the independent components can be estimated, because any constant multiplying an independent component in eq. (1) could be cancelled by dividing the corresponding column of the mixing matrix $\mathbf{A}$ by the same constant. For mathematical convenience, we define here that the independent components s_i have unit variance. This makes the (non-Gaussian) independent components unique, up to their signs. Note that no order is defined between the independent components.

In *blind source separation* (Cardoso, 1990; Jutten and Herault, 1991), the observed values of $\mathbf{v}$ correspond to a realization of an m-dimensional discrete-time signal $\mathbf{v}(t)$, $t = 1, 2, ...$ Then the components $s_i(t)$ are called source signals, which are usually original, uncorrupted signals or noise sources.

Another possible application of ICA is *feature extraction* (Bell and Sejnowski, 1996a; Bell and Sejnowski, 1996b; Hurri et al., 1996). Then the columns of $\mathbf{A}$ represent features, and s_i signals the presence and the 'amplitude' of the i-th feature in the observed data $\mathbf{v}$.

The problem of estimating the matrix $\mathbf{A}$ in eq. (1) can be somewhat simplified by performing a preliminary *sphering* or prewhitening of the data $\mathbf{v}$ (Cardoso, 1990; Comon, 1994; Oja and Karhunen, 1995). The observed vector $\mathbf{v}$ is linearly transformed to a vector $\mathbf{x} = \mathbf{M}\mathbf{v}$ such that its elements x_i are mutually uncorrelated and all have unit variance. Thus the correlation matrix of $\mathbf{x}$ equals unity: $E\{\mathbf{x}\mathbf{x}^T\} = \mathbf{I}$. This transformation is always possible and can be accomplished by classical Principal Component Analysis. At the same time, the dimensionality of the data should be reduced so that the dimension of the transformed data vector $\mathbf{x}$ equals n, the number of independent components. This also has the effect of reducing noise. After the transformation we have

$$\mathbf{x} = \mathbf{M}\mathbf{v} = \mathbf{M}\mathbf{A}\mathbf{s} = \mathbf{B}\mathbf{s} \tag{2}$$

where $\mathbf{B} = \mathbf{M}\mathbf{A}$ is an *orthogonal* matrix due to our assumptions on the components s_i: it holds $E\{\mathbf{x}\mathbf{x}^T\} = \mathbf{B}E\{\mathbf{s}\mathbf{s}^T\}\mathbf{B}^T = \mathbf{B}\mathbf{B}^T = \mathbf{I}$. Thus we have reduced the problem of finding an arbitrary full-rank matrix $\mathbf{A}$ to the simpler problem of finding an orthogonal matrix $\mathbf{B}$, which then gives $\mathbf{s} = \mathbf{B}^T\mathbf{x}$. If the i-th column of $\mathbf{B}$ is denoted $\mathbf{b}_i$, then the i-th independent component can be computed from the observed $\mathbf{x}$ as $s_i = (\mathbf{b}_i)^T\mathbf{x}$.

The current algorithms for Independent Component Analysis can be roughly divided into two categories. The algorithms in the first category (Cardoso, 1992; Comon, 1994) rely on batch computations minimizing or maximizing some relevant criterion functions. The problem with these algorithms is that they require very complex matrix or tensorial operations. The second category contains adaptive algorithms often based on stochastic gradient methods, which may have implementations in neural networks (Amari et al., 1996; Bell and Sejnowski, 1995; Delfosse and Loubaton, 1995; Hyvärinen and Oja, 1996; Jutten and Herault, 1991; Moreau and Macchi, 1993; Oja and Karhunen, 1995). The main problem with this category is the slow convergence, and the fact that the convergence depends crucially on the correct choice of the learning rate parameters.

In this paper we introduce a novel approach for performing the computations needed in ICA[1]. We introduce an algorithm using a very simple, yet highly efficient, *fixed-point iteration scheme* for finding the local extrema of the kurtosis of a linear combination of the observed variables. It is well-known (Delfosse and Loubaton, 1995) that finding the local extrema of kurtosis is equivalent to estimating the non-Gaussian independent components. However, the convergence of our algorithm will be proven independently of these well-known results. The computations can be performed either in batch mode or semi-adaptively.

The new algorithm is introduced and analyzed in Section 3, after presenting in Section 2 a short review of kurtosis minimization/maximization and its relation to neural network type learning rules.

2 ICA by Kurtosis Minimization and Maximization

Most suggested solutions to the ICA problem use the fourth-order cumulant or *kurtosis* of the signals, defined for a zero-mean random variable v as

$$\mathrm{kurt}(v) = E\{v^4\} - 3(E\{v^2\})^2 \tag{3}$$

For a Gaussian random variable, kurtosis is zero; for densities peaked at zero, it is positive, and for flatter densities, negative. Note that for two independent random variables v_1 and v_2 and for a scalar α, it holds $\mathrm{kurt}(v_1 + v_2) = \mathrm{kurt}(v_1) + \mathrm{kurt}(v_2)$ and $\mathrm{kurt}(\alpha v_1) = \alpha^4 \,\mathrm{kurt}(v_1)$.

Let us search for a linear combination of the sphered observations x_i, say, $\mathbf{w}^T\mathbf{x}$, such that it has maximal or minimal kurtosis. Obviously, this is meaningful only if the norm of $\mathbf{w}$ is somehow bounded; let us assume $\|\mathbf{w}\| = 1$. Using the orthogonal mixing matrix $\mathbf{B}$, let us define $\mathbf{z} = \mathbf{B}^T\mathbf{w}$. Then also $\|\mathbf{z}\| = 1$. Using eq. (2) and the properties of the kurtosis, we have

$$\mathrm{kurt}(\mathbf{w}^T\mathbf{x}) = \mathrm{kurt}(\mathbf{w}^T\mathbf{B}\mathbf{s}) = \mathrm{kurt}(\mathbf{z}^T\mathbf{s}) = \sum_{i=1}^{n} z_i^4 \,\mathrm{kurt}(s_i) \tag{4}$$

Under the constraint $\|\mathbf{w}\| = \|\mathbf{z}\| = 1$, the function (4) has a number of local minima and maxima. For simplicity, let us assume for the moment that the mixture contains at least one independent component whose kurtosis is negative, and at least one whose kurtosis is positive. Then, as was shown by Delfosse and Loubaton (1995), the extremal points of (4) are the canonical base vectors $\mathbf{z} = \pm\mathbf{e}_j$, i.e. vectors whose

[1]It was brought to our attention that a similar algorithm for blind deconvolution was proposed by Shalvi and Weinstein (1993).

all components are zero except one component which equals ± 1. The corresponding weight vectors are $\mathbf{w} = \mathbf{B}\mathbf{z} = \mathbf{B}\mathbf{e}_j = \mathbf{b}_j$ (perhaps with a minus sign), i.e. the columns of the orthogonal mixing matrix $\mathbf{B}$. So, by minimizing or maximizing the kurtosis in eq. (4) under the given constraint, the columns of the mixing matrix are obtained as solutions for $\mathbf{w}$, and the linear combination itself will be one of the independent components: $\mathbf{w}^T\mathbf{x} = (\mathbf{b}_i)^T\mathbf{x} = s_i$. Equation (4) also shows that Gaussian components cannot be estimated by this way, because for them $\mathrm{kurt}(s_i)$ is zero.

To actually minimize or maximize $\mathrm{kurt}(\mathbf{w}^T\mathbf{x})$, a neural algorithm based on gradient descent or ascent can be used (Delfosse and Loubaton, 1995; Hyvärinen and Oja, 1996). Then $\mathbf{w}$ is interpreted as the weight vector of a neuron with input vector $\mathbf{x}$. The objective function can be simplified because the inputs have been sphered: it holds

$$\mathrm{kurt}(\mathbf{w}^T\mathbf{x}) = E\{(\mathbf{w}^T\mathbf{x})^4\} - 3[E\{(\mathbf{w}^T\mathbf{x})^2\}]^2 = E\{(\mathbf{w}^T\mathbf{x})^4\} - 3\|\mathbf{w}\|^4 \tag{5}$$

Also the constraint $\|\mathbf{w}\| = 1$ must be taken into account, e.g. by a penalty term (Hyvärinen and Oja, 1996). Then the final objective function is

$$J(\mathbf{w}) = E\{(\mathbf{w}^T\mathbf{x})^4\} - 3\|\mathbf{w}\|^4 + F(\|\mathbf{w}\|^2) \tag{6}$$

where F is a penalty term due to the constraint. Several forms for the penalty term have been suggested by Hyvärinen and Oja (1996)[2]. In the following, the exact form of F is not important. Denoting by $\mathbf{x}(t)$ the sequence of observations, by $\mu(t)$ the learning rate sequence, and by f the derivative of $F/2$, the on-line learning algorithm then has the form

$$\mathbf{w}(t+1) = \mathbf{w}(t) \pm \mu(t)[\mathbf{x}(t)(\mathbf{w}(t)^T\mathbf{x}(t))^3 - 3\|\mathbf{w}(t)\|^2\mathbf{w}(t) + f(\|\mathbf{w}(t)\|^2)\mathbf{w}(t)] \tag{7}$$

The first two terms in brackets are obtained from the gradient of $\mathrm{kurt}(\mathbf{w}^T\mathbf{x})$ when instantaneous values are used instead of the expectation. The third term in brackets is obtained from the gradient of $F(\|\mathbf{w}\|^2)$; note that as long as this is a function of $\|\mathbf{w}\|^2$ only, its gradient has the form *scalar* $\times\, \mathbf{w}$. Positive sign before the brackets means finding the local maxima, negative sign corresponds to local minima.

The convergence of this kind of algorithms can be proven using the principles of stochastic approximation. The advantage of such neural learning rules is that the inputs $\mathbf{x}(t)$ can be used in the algorithm at once, thus enabling fast adaptation in a non-stationary environment. A resulting trade-off, however, is that the convergence is slow, and depends on a good choice of the learning rate sequence $\mu(t)$. A bad choice of the learning rate can, in practice, destroy convergence. Therefore, some ways to make the learning radically faster and more reliable may be needed. The fixed-point iteration algorithms are such an alternative.

The fixed points $\mathbf{w}$ of the learning rule (7) are obtained by taking the expectations and equating the change in the weight to 0:

$$E\{\mathbf{x}(\mathbf{w}^T\mathbf{x})^3\} - 3\|\mathbf{w}\|^2\mathbf{w} + f(\|\mathbf{w}\|^2)\mathbf{w} = 0 \tag{8}$$

[2] Note that in (Hyvärinen and Oja, 1996), the second term in J was also included in the penalty term.

The time index t has been dropped. A deterministic iteration could be formed from eq. (8) by a number of ways, e.g. by standard numerical algorithms for solving such equations. A very fast iteration is obtained, as shown in the next section, if we write (8) in the form

$$\mathbf{w} = scalar \times (E\{\mathbf{x}(\mathbf{w}^T\mathbf{x})^3\} - 3\|\mathbf{w}\|^2\mathbf{w}) \tag{9}$$

Actually, because the norm of $\mathbf{w}$ is irrelevant, it is the direction of the right hand side that is important. Therefore the *scalar* in eq. (9) is not significant and its effect can be replaced by explicit normalization, or the projection of $\mathbf{w}$ onto the unit sphere.

3 Fixed-Point Algorithm

3.1 The Algorithm

3.1.1 Estimating one independent component

Assume that we have collected a sample of the sphered (or prewhitened) random vector $\mathbf{x}$, which in the case of blind source separation is a collection of linear mixtures of independent source signals according to eq. (2). Using the derivation of the preceding section, we get the following *fixed-point algorithm for ICA:*

1. Take a random initial vector $\mathbf{w}(0)$ of norm 1. Let $k = 1$.
2. Let $\mathbf{w}(k) = E\{\mathbf{x}(\mathbf{w}(k-1)^T\mathbf{x})^3\} - 3\mathbf{w}(k-1)$. The expectation can be estimated using a large sample of $\mathbf{x}$ vectors (say, 1,000 points).
3. Divide $\mathbf{w}(k)$ by its norm.
4. If $|\mathbf{w}(k)^T\mathbf{w}(k-1)|$ is not close enough to 1, let $k = k + 1$ and go back to step 2. Otherwise, output the vector $\mathbf{w}(k)$.

The final vector $\mathbf{w}(k)$ given by the algorithm equals one of the columns of the (orthogonal) mixing matrix $\mathbf{B}$. In the case of blind source separation, this means that $\mathbf{w}(k)$ separates *one* of the non-Gaussian source signals in the sense that $\mathbf{w}(k)^T\mathbf{x}(t)$, $t = 1, 2, ...$ equals one of the source signals.

A remarkable property of our algorithm is that a very small number of iterations, usually 5-10, seems to be enough to obtain the maximal accuracy allowed by the sample data. This is due to the cubic convergence shown below.

3.1.2 Estimating several independent components

To estimate n independent components, we run this algorithm n times. To ensure that we estimate each time a different independent component, we only need to add a simple *orthogonalizing projection* inside the loop. Recall that the columns of the mixing matrix $\mathbf{B}$ are orthonormal because of the sphering. Thus we can estimate the independent components one by one by projecting the current solution $\mathbf{w}(k)$ on the space orthogonal to the columns of the mixing matrix $\mathbf{B}$ previously found. Define the matrix $\overline{\mathbf{B}}$ as a matrix whose columns are the previously found columns of $\mathbf{B}$. Then add the projection operation in the beginning of step 3:

3. Let $\mathbf{w}(k) = \mathbf{w}(k) - \overline{\mathbf{B}}\,\overline{\mathbf{B}}^T\mathbf{w}(k)$. Divide $\mathbf{w}(k)$ by its norm.

Also the initial random vector should be projected this way before starting the iterations. To prevent estimation errors in $\overline{\mathbf{B}}$ from deteriorating the estimate $\mathbf{w}(k)$, this projection step can be omitted after the first few iterations: once the solution $\mathbf{w}(k)$ has entered the basin of attraction of one of the fixed points, it will stay there and converge to that fixed point.

In addition to the hierarchical (or sequential) orthogonalization described above, any other method of orthogonalizing the weight vectors could also be used. In some applications, a *symmetric orthogonalization* might be useful. This means that the fixed-point step is first performed for all the n weight vectors, and then the matrix $\mathbf{W}(k) = (\mathbf{w}_1(k), ..., \mathbf{w}_n(k))$ of the weight vectors is orthogonalized, e.g., using the well-known formula

$$\text{Let } \mathbf{W}(k) = \mathbf{W}(k)(\mathbf{W}(k)^T\mathbf{W}(k))^{-1/2} \tag{10}$$

where $(\mathbf{W}(k)^T\mathbf{W}(k))^{-1/2}$ is obtained from the eigenvalue decomposition of $\mathbf{W}(k)^T\mathbf{W}(k) = \mathbf{EDE}^T$ as $(\mathbf{W}(k)^T\mathbf{W}(k))^{-1/2} = \mathbf{ED}^{-1/2}\mathbf{E}^T$. However, the convergence proof below applies only to the hierarchical orthogonalization.

3.1.3 A semi-adaptive version

A disadvantage of many batch algorithms is that large amounts of data must be stored simultaneously in working memory. Our fixed-point algorithm, however, can be used in a *semi-adaptive manner* so as to avoid this problem. This can simply be accomplished by computing the expectation $E\{\mathbf{x}(\mathbf{w}(k-1)^T\mathbf{x})^3\}$ by an on-line algorithm for N consecutive sample points, keeping $\mathbf{w}(k-1)$ fixed, and updating the vector $\mathbf{w}(k)$ after the average over all the N sample points has been computed.

This semi-adaptive version also makes adaptation to non-stationary data possible. Thus the semi-adaptive algorithm combines many of the advantages usually attributed to either on-line or batch algorithms.

3.2 Convergence Proof

Now we prove the convergence of our algorithm. To begin with, make the change of variables $\mathbf{z}(k) = \mathbf{B}^T\mathbf{w}(k)$. Note that the effect of the projection step is to set to zero all components $z_i(k)$ of $\mathbf{z}(k)$ such that the i-th column of $\mathbf{B}$ has already been estimated. Therefore, we can simply consider in the following the algorithm without the projection step, taking into account that the $z_i(k)$ corresponding to the independent components previously estimated, are zero.

First, using (2), we get the following form for step 2 of the algorithm:

$$\mathbf{z}(k) = E\{\mathbf{s}(\mathbf{z}(k-1)^T\mathbf{s})^3\} - 3\mathbf{z}(k-1) \tag{11}$$

Expanding the first term, we can calculate explicitly the expectation, and obtain for the i-th component of the vector $\mathbf{z}(k)$:

$$z_i(k) = E\{s_i^4\}z_i(k-1)^3 + 3\sum_{j\neq i} z_j(k-1)^2 z_i(k-1) - 3z_i(k-1) \tag{12}$$

where we have used the fact that by the statistical independence of the s_i, we have $E\{s_i^2 s_j^2\} = 1$, and $E\{s_i^3 s_j\} = E\{s_i^2 s_j s_l\} = E\{s_i s_j s_l s_m\} = 0$ for four different indices i,j,l, and m. Using $\|\mathbf{z}(k)\| = \|\mathbf{w}(k)\| = 1$, eq. (12) simplifies to

$$z_i(k) = \operatorname{kurt}(s_i)\; z_i(k-1)^3 \tag{13}$$

where $\operatorname{kurt}(s_i) = E\{s_i^4\} - 3$ is the kurtosis of the i-th independent component. Note that the subtraction of $3\mathbf{w}(k-1)$ from the right side cancelled the term due to the cross-variances, enabling direct access to the fourth-order cumulants. Choosing j so that $\operatorname{kurt}(s_j) \neq 0$ and $z_j(k-1) \neq 0$, we further obtain

$$\frac{|z_i(k)|}{|z_j(k)|} = \frac{|\operatorname{kurt}(s_i)|}{|\operatorname{kurt}(s_j)|}\left(\frac{|z_i(k-1)|}{|z_j(k-1)|}\right)^3 \tag{14}$$

Note that the assumption $z_j(k-1) \neq 0$ implies that the j-th column of $\mathbf{B}$ is not among those already found. Next note that $|z_i(k)|/|z_j(k)|$ is not changed by the normalization step 3. It is therefore possible to solve explicitly $|z_i(k)|/|z_j(k)|$ from this recursive formula, which yields

$$\frac{|z_i(k)|}{|z_j(k)|} = \frac{\sqrt{|\operatorname{kurt}(s_j)|}}{\sqrt{|\operatorname{kurt}(s_i)|}}\left(\frac{\sqrt{|\operatorname{kurt}(s_i)|}\,|z_i(0)|}{\sqrt{|\operatorname{kurt}(s_j)|}\,|z_j(0)|}\right)^{3^k} \tag{15}$$

for all $k > 0$. For $j = \arg\max_p \sqrt{|\operatorname{kurt}(s_p)|}\,|z_p(0)|$, we see that all the other components $z_i(k),\ i \neq j$ quickly become small compared to $z_j(k)$. Taking the normalization $\|\mathbf{z}(k)\| = \|\mathbf{w}(k)\| = 1$ into account, this means that $z_j(k) \to 1$ and $z_i(k) \to 0$ for all $i \neq j$. This implies that $\mathbf{w}(k) = \mathbf{B}\mathbf{z}(k)$ converges to the column $\mathbf{b}_j$ of the mixing matrix $\mathbf{B}$, for which the kurtosis of the corresponding independent component s_j is not zero, and which has not yet been found. This proves the convergence of our algorithm.

4 Simulation Results

The fixed-point algorithm was applied to blind separation of 4 source signals from 4 observed mixtures. Two of the source signals had a uniform distribution, and the other two were obtained as cubes of Gaussian variables. Thus, the source signals included both sub-Gaussian and super-Gaussian signals. Using different random mixing matrices and initial values $\mathbf{w}(0)$, five iterations were usually sufficient for estimation of one column of the orthogonal mixing matrix to an accuracy of 4 decimal places.

Next, the convergence speed of our algorithm was compared with the speed of the corresponding neural stochastic gradient algorithm. In the neural algorithm, an empirically optimized learning rate sequence was used. The computational overhead of the fixed-point algorithm was optimized by initially using a small sample size (200 points) in step 2 of the algorithm, and increasing it at every iteration for greater accuracy. The number of floating point operations was calculated for both methods.

The fixed-point algorithm needed only 10 per cent of the floating point operations required by the neural algorithm. Note that this result was achieved with an empirically optimized learning rate. If we had been obliged to choose the learning rate without preliminary testing, the speed-up factor would have been of the order of 100. In fact, the neural stochastic gradient algorithm might not have converged at all.

These simulation results confirm the theoretical implications of very fast convergence.

5 Discussion

We introduced a batch version of a neural learning algorithm for Independent Component Analysis. This algorithm, which is based on the fixed-point method, has several advantages as compared to other suggested ICA methods.

1. Equation (15) shows that the convergence of our algorithm is *cubic*. This means very fast convergence and is rather unique among the ICA algorithms. It is also in contrast to other similar fixed-point algorithms, like the power method, which often have only linear convergence. In fact, our algorithm can be considered a higher-order generalization of the power method for tensors.

2. Contrary to gradient-based algorithms, there is no learning rate or other adjustable parameters in the algorithm, which makes it easy to use, and more reliable.

3. The algorithm, in its hierarchical version, finds the independent components one at a time, instead of working in parallel like most of the suggested ICA algorithms that solve the entire mixing matrix. This makes it possible to estimate only certain desired independent components, provided we have sufficient prior information of the weight matrices corresponding to those components. For example, if the initial weight vector of the algorithm is the first principal component, the algorithm finds probably the most important independent component, i.e. such that the norm of the corresponding column in the original mixing matrix $\mathbf{A}$ is the largest.

4. Both components of negative kurtosis (i.e. sub-Gaussian components) and components of positive kurtosis (i.e. super-Gaussian components) can be found, by starting the algorithm from different

initial points, and possibly removing the effect of the already found independent components by a projection onto an orthogonal subspace. If just one of the independent components is Gaussian (or otherwise has zero kurtosis), it can be estimated as the residual that is left over after extracting all other independent components. Recall that more than one Gaussian independent component cannot be estimated in the ICA model.

5. Although the algorithm was motivated above as a short-cut method to make neural learning for kurtosis minimization / maximization faster, its convergence was proven independently of the neural algorithm and the well-known results on the connection between ICA and kurtosis. Indeed, our proof is based on principles different from those used so far in ICA, and thus opens new lines for research.

Recent developments of the theory presented in this paper can be found in (Hyvärinen, 1997a; Hyvärinen, 1997b).

References

Amari, S., Cichocki, A., and Yang, H. (1996). A new learning algorithm for blind source separation. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing 8 (Proc. NIPS'95)*, pages 757–763. MIT Press, Cambridge, MA.

Bell, A. and Sejnowski, T. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159.

Bell, A. and Sejnowski, T. (1996a). The 'independent components' of natural scenes are edge filters. *Vision Research*. To appear.

Bell, A. and Sejnowski, T. (1996b). Learning higher-order structure of a natural sound. *Network*, 7:261–266.

Cardoso, J.-F. (1990). Eigen-structure of the fourth-order cumulant tensor with application to the blind source separation problem. In *Proc. ICASSP'90*, pages 2655–2658, Albuquerque, NM, USA.

Cardoso, J.-F. (1992). Iterative techniques for blind source separation using only fourth-order cumulants. In *Proc. EUSIPCO*, pages 739–742, Brussels, Belgium.

Comon, P. (1994). Independent component analysis – a new concept? *Signal Processing*, 36:287–314.

Delfosse, N. and Loubaton, P. (1995). Adaptive blind separation of independent sources: a deflation approach. *Signal Processing*, 45:59–83.

Hurri, J., Hyvärinen, A., Karhunen, J., and Oja, E. (1996). Image feature extraction using independent component analysis. In *Proc. NORSIG'96*, pages 475–478, Espoo, Finland.

Hyvärinen, A. (1997a). A family of fixed-point algorithms for independent component analysis. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'97)*, pages 3917–3920, Munich, Germany.

Hyvärinen, A. (1997b). Independent component analysis by minimization of mutual information. Technical report, Helsinki University of Technology, Laboratory of Computer and Information Science.

Hyvärinen, A. and Oja, E. (1996). A neuron that learns to separate one independent component from linear mixtures. In *Proc. IEEE Int. Conf. on Neural Networks*, pages 62–67, Washington, D.C.

Jutten, C. and Herault, J. (1991). Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10.

Moreau, E. and Macchi, O. (1993). New self-adaptive algorithms for source separation based on contrast functions. In *Proc. IEEE Signal Processing Workshop on Higher Order Statistics*, pages 215–219, Lake Tahoe, USA.

Oja, E. and Karhunen, J. (1995). Signal separation by nonlinear hebbian learning. In Palaniswami, M., Attikiouzel, Y., Marks, R., Fogel, D., and Fukuda, T., editors, *Computational Intelligence - a Dynamic System Perspective*, pages 83 – 97. IEEE Press, New York.

Shalvi, O. and Weinstein, E. (1993). Super-exponential methods for blind deconvolution. *IEEE Trans. on Information Theory*, 39(2):504:519.